ENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1993	Professional Paper

4. TITLE AND SUBTITLE

SOFTWARE IS A PRODUCT ... NOT!

5. FUNDING NUMBERS

In House Funding

6. AUTHOR(S)

M. D. Shapiro

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Command, Control and Ocean Surveillance Center (NCCOSC)
RDT&E Division
San Diego, CA 92152–5001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Command, Control and Ocean Surveillance Center (NCCOSC)
RDT&E Division
San Diego, CA 92152–5001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION

13. ABSTRACT (Maximum 200 words)

Much of our planning for software uses the hardware product model. While we're not perfect, we do a fairly good job of managing hardware engineering for required time, quality, and money. But software more often than not is late, doesn't do what we want it to do, and costs too much. And that's a problem.

We can't see a solution to our problem—perhaps because we have made the solution invisible by thinking about it in the wrong terms.

I propose a vocabulary change. We will better understand the process if we consider software as a *service*, not a *product*. Let me expand on this statement.

I do not believe we must do any of the software-building activities differently. Instead, from the perspective of scheduling, budgeting, and delivering software, we should use the service paradigm rather than the product paradigm.

Published in *IEEE Computer Magazine*, Vol. 26, No. 9, Sep 1992, p 128.

14. SUBJECT TERMS	15. NUMBER OF PAGES
scheduling budgeting	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT

21a. NAME OF RESPONSIBLE INDIVIDUAL	21b. TELEPHONE *(include Area Code)*	21c. OFFICE SYMBOL
M. D. Shapiro	(619) 553–4080	Code 411

from Computer, September 1992, vol. 25, no. 9

"Any clod can have the facts, but having opinions is an art."
Charles McCabe, San Francisco Chronicle

THE OPEN CHANNEL

Editor: Will Tracz, IBM, Federal Systems Company, Mail Drop 0210, Owego, NY 13827
e-mail tracz@owgvm0.vnet.ibm.com

Software is a product . . . NOT!

Much of our planning for software uses the hardware product model. While we're not perfect, we do a fairly good job of managing hardware engineering for required time, quality, and money. But software more often than not is late, doesn't do what we want it to do, and costs too much. And that's a problem.

We can't see a solution to our problem — perhaps because we have made the solution invisible by thinking about it in the wrong terms.

I propose a vocabulary change. We will better understand the process if we consider software as a *service*, not a *product*. Let me expand on this statement.

I do not believe we must do any of the software-building activities differently. Instead, from the perspective of scheduling, budgeting, and delivering software, we should use the service paradigm rather than the product paradigm.

Hardware versus software. We've all pondered the ways in which software differs from hardware:

- After initial development, most hardware cost is in manufacturing. Software has a small manufacturing cost.
- Hardware has physical parts that wear out, but can be replaced with identical or improved parts; the physical environment seldom changes. Software parts do not wear out, but often need replacement with greatly improved parts for changing environmental requirements.
- After initial development, hardware requires few engineering personnel for product maintenance. Software needs many highly skilled software engineers.
- Hardware is difficult and expensive to change in the field. Software can be upgraded easily and inexpensively in the field.
- Hardware engineering has matured into a craft. Software engineering still has a long way to go.

Hardware matches our notion of a product. But does software? Can we do better with a service paradigm?

Software as a service. What is a service paradigm? Service is intangible. Software provides a delivery system for results. Users of our software do not care what program they have for a word processor, spreadsheet, or graphics editor. They want results: a proposal, a financial statement, or a presentation slide.

A service is successful when the customer wants to continue using it. Under my proposal, our emphasis would change from building and maintaining a product to providing the service users need to do their jobs.

We must plan and budget our efforts toward reaching this goal. And we should teach customers this same attitude for acquiring and using software.

Upgrades as service. When the software industry began, everything was free. Manufacturers gave away software to sell machines. Users freely shared software. Eventually, we realized we could make money from software by treating it as a product. Companies began charging for software, although they seldom sold it. Instead they leased or licensed it for a fee, often with a maintenance charge.

As personal computers came along, software truly became a product. Now, we can go into a store and buy software off the shelf. We own it (no matter what the "license agreement" says) and can use it forever without ever paying any more.

But computer hardware capabilities keep increasing and we demand software improvements. Competitors appear, selling fancier versions of products. Companies enhance their products and sell "upgrades" through the mail.

With much fanfare, Microsoft started selling its 5.0 upgrade to MS-DOS in stores. Soon, most firms were selling upgrades — not only to their own products, but also as replacements for their competitors' products. Upgrades appeared on the best-selling software lists. Customers became accustomed to (but not necessarily happy with) paying for a product and, at intervals, its upgrades. The pricing scheme began to resemble that of a service like a book or record club.

The next stage? Because companies complained about the need to continually purchase upgrades, Microsoft announced that it would begin an upgrade subscription service — much like a magazine subscription.

So we've moved from software as a free service to software as a paid subscription service. In all these modes, we call it a product. But lurking in the background is that idea of software as a service, not a product. And if we plan and manage it as such, we might have a better hold on planning, scheduling, and performing its activities.

Avoiding maintenance. An interesting thing happens when software is no longer a product. We no longer maintain it.

We've often said that software maintenance differs from hardware maintenance. Instead of keeping parts that wear out from failing, software maintenance now embodies fixing errors and enhancing software for new requirements or environments. But if software becomes a service, we emphasize adjustments for flaws as they appear and expansion to new and improved facilities.

What's the difference? It's a matter of attitude. With the attitude that software is a continuing service, we plan and budget for what was called maintenance as part of that service instead of tacking it on as a necessary, but often maligned, phase at the end of the product development process.

Will it work? Can this change of vocabulary make a difference? *No*, if we merely apply a new word to old ideas. But *yes*, if we allow it to change our attitudes toward the way we think about, plan, and do software.

Michael D. Shapiro
Computer scientist
San Diego, California
mshapiro@nosc.mil